

EBook Gratis

APRENDIZAJE wxpython

Free unaffiliated eBook created from **Stack Overflow contributors.**

#wxpython

Tabla de contenido

Acerca de

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: wxpython

It is an unofficial and free wxpython ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official wxpython.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

Capítulo 1: Empezando con wxpython

Observaciones

¿Qué es wxPython

En pocas palabras, wxPython es un conjunto de enlaces a la biblioteca de la GUI multiplataforma de C ++ de **wxWidgets** .

Ok que es wxwidgets

La biblioteca wxWidgets proporciona un conjunto gratuito, gratuito y de código abierto de abstracciones para los diversos elementos de la GUI para que los controles nativos se sigan utilizando, cuando estén disponibles, manteniendo el aspecto, la sensación y la velocidad nativos. Como tal, proporciona una abstracción para la creación de GUI y una serie de otras utilidades en una plataforma que permite a los desarrolladores crear aplicaciones para Windows, Mac OS X, Linux y otras plataformas utilizando un solo código base. wxWidgets se inició en 1992 y puede ver un historial detallado aquí . La biblioteca wxWidgets se distribuye bajo la licencia wxWindows, que se basa en la **L-GPL pero con una cláusula de excepción** . *La cláusula de excepción le permite vincular su aplicación de forma dinámica o estática a wxWidgets* ***sin*** *el requisito de distribuir la fuente para su propia aplicación. En otras palabras, puede usar wxWidgets para proyectos* ***gratuitos o comerciales*** *,* ***sin costo alguno*** *. La licencia lo alienta a devolver las mejoras que realice a la propia biblioteca wxWidgets.*

Los aspectos destacados, *tenga en cuenta que wxWidgets comprende 100s de clases para el desarrollo de aplicaciones multiplataforma* :

- Diseño de la ventana usando Sizers
- Contextos de dispositivos (junto con bolígrafos, pinceles y fuentes)
- Sistema integral de gestión de eventos
- Visor de ayuda HTML
- Reproducción de sonido y video
- Soporte Unicode e Internacionalización.
- Documentar / Ver Arquitectura
- Archiecture de impresión
- Enchufes
- Multihilo
- Manipulación de archivos y directorios
- Ayuda en línea y sensible al contexto
- Procesamiento de HTML
- Contenedores básicos
- Carga de imágenes, guardado, dibujo y manipulación.
- Biblioteca de fecha y hora y temporizadores

- Manejo de errores
- Portapapeles y arrastrar y soltar

Tenga en cuenta que algunas de estas funciones, *por ejemplo* , los *subprocesos,* no están realmente relacionados con la GUI pero proporcionan una abstracción multiplataforma útil para que, en el caso de los subprocesos, un conjunto de códigos de aplicación funcione en cualquier plataforma compatible.

Durante muchos años, la biblioteca wxWidgets produjo 4 compilaciones separadas, *además de depurar compilaciones a* partir de un conjunto de bibliotecas estáticas y dinámicas de código fuente creadas para ASCII y Unicode. Por lo general, está disponible pre-construido en las variantes más comunes y como código fuente para compilar *con las diversas opciones* para el entorno de destino y con la cadena de herramientas C ++ de desarrolladores con el apoyo de numerosas cadenas de herramientas.

Los enlaces de python para esta biblioteca y algunas adiciones forman wxPython.

Volver a ¿Qué es wxPython, (qué me da)?

wxPython le da a un desarrollador una forma de beneficiarse de una biblioteca de GUI multiplataforma, con una licencia clara, al mismo tiempo que brinda los beneficios de Python. Al igual que wxWidgets y Python, wxPython es gratuito, gratuito y de código abierto, y está disponible para su uso y distribución en proyectos tanto gratuitos como comerciales *sin el requisito de distribuir su código fuente* .

- Suite GUI completa que incluye (pero no se limita a):
  - Windows (incluyendo Windows MDI)
  - Magos
  - Marcos y MiniFrames
  - Diálogos, Estándar, Avanzado y Personalizado.
  - Libros, árboles, cuadrículas y controles de vista de datos
  - Indicadores, controles deslizantes, giradores, animaciones, portapapeles, arrastrar y soltar
  - Compatibilidad con HTML, PDF y visor de imágenes
  - Los componentes de la GUI se pueden posicionar de forma absoluta, pero se recomienda encarecidamente utilizar un diseño basado en el medidor que admita el tamaño automático, etc.
- Plataforma cruzada: soporte GUIs para Windows, OS-X y Linux con una base de código única *sin declaraciones condicionales en su código*
- Velocidad nativa, look & feel.
- Prototipo rápido, prueba y depuración: *recuerda que esto es Python*
- Ejecutar y editar muestras de casi todo en el paquete de demostración.
- Licencia clara para uso gratuito incluso en productos comerciales.
- Si es necesario, su GUI de python puede volver a ajustarse a una GUI de WxWidgets de C ++ más adelante, ya que ya la está utilizando.
- Comunidad de usuarios y desarrolladores grande, activa y útil tanto en StackOverflow como

en listas de correo .

Tenga en cuenta que donde Python proporciona un mecanismo multiplataforma para implementar las funciones de utilidad de wxWidgets, una *vez más el subproceso es un buen ejemplo* , se omite **intencionalmente** en wxPython.

wxPython también tiene un conjunto muy grande de demostraciones que se pueden ejecutar, probar y editar desde el paquete Documentos y Demo.

Sabores de wxPython

ASCII vs Unicode :

Durante muchos años, al *igual que con wxWidgets* , los desarrolladores tuvieron que elegir entre compilaciones ASCII y Unicode, así como también necesitar una compilación para su versión específica de python, así como las opciones de 32/64 bits. A partir de aproximadamente wxPython 2.8.9, la versión ASCII solo de wxPython se ha eliminado, por lo que la compatibilidad con Unicode siempre está disponible.

Clásico vs. Phoenix :

Desde wxPython 3.0.0 ha existido la compilación "Clásica" *lanzada* de wxPython y una compilación de Phoenix *actualmente no lanzada* . La compilación clásica tiende a retrasarse con respecto a las compilaciones de wxWidgets de los mismos números y el paquete de documentación es C ++: está disponible para su descarga en varias plataformas (consulte Instalación de Classic), en el caso de Windows como instalador ejecutable. Los enlaces de Phoenix, que se generan en gran parte de forma automática, deberían seguir más de cerca las compilaciones wxWidgets y también incluir la documentación específica de wxPython: es compilable desde la fuente o compilaciones nocturnas, *ya que las ruedas* se pueden obtener usando **pip** , (ver Instalación de Phoenix).

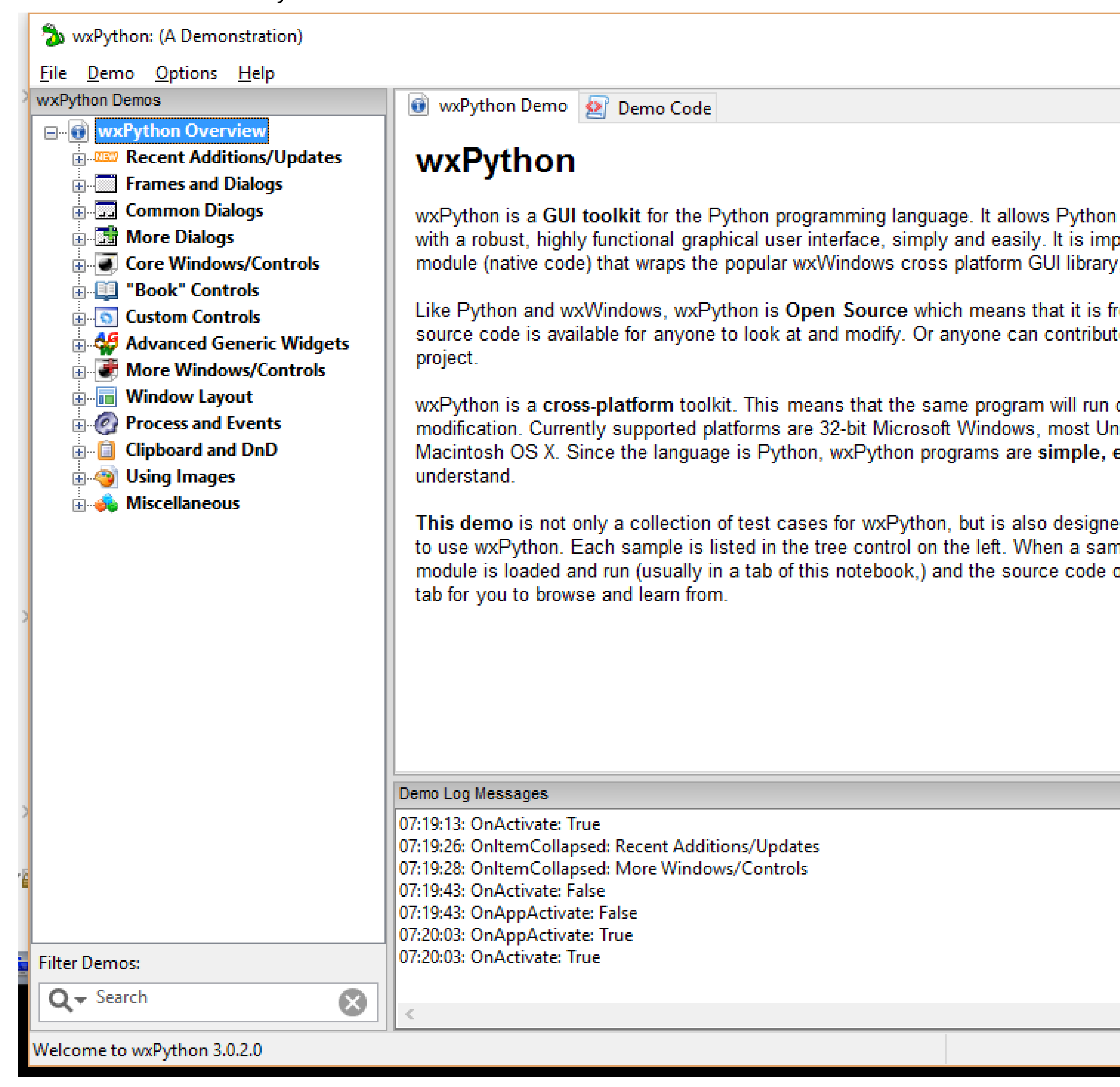
En wxPython pero no wxWidgets

wxPython amplía la biblioteca wxWidgets con una serie de características, *las siguientes son solo algunas de las* que no están disponibles en wxWidgets:

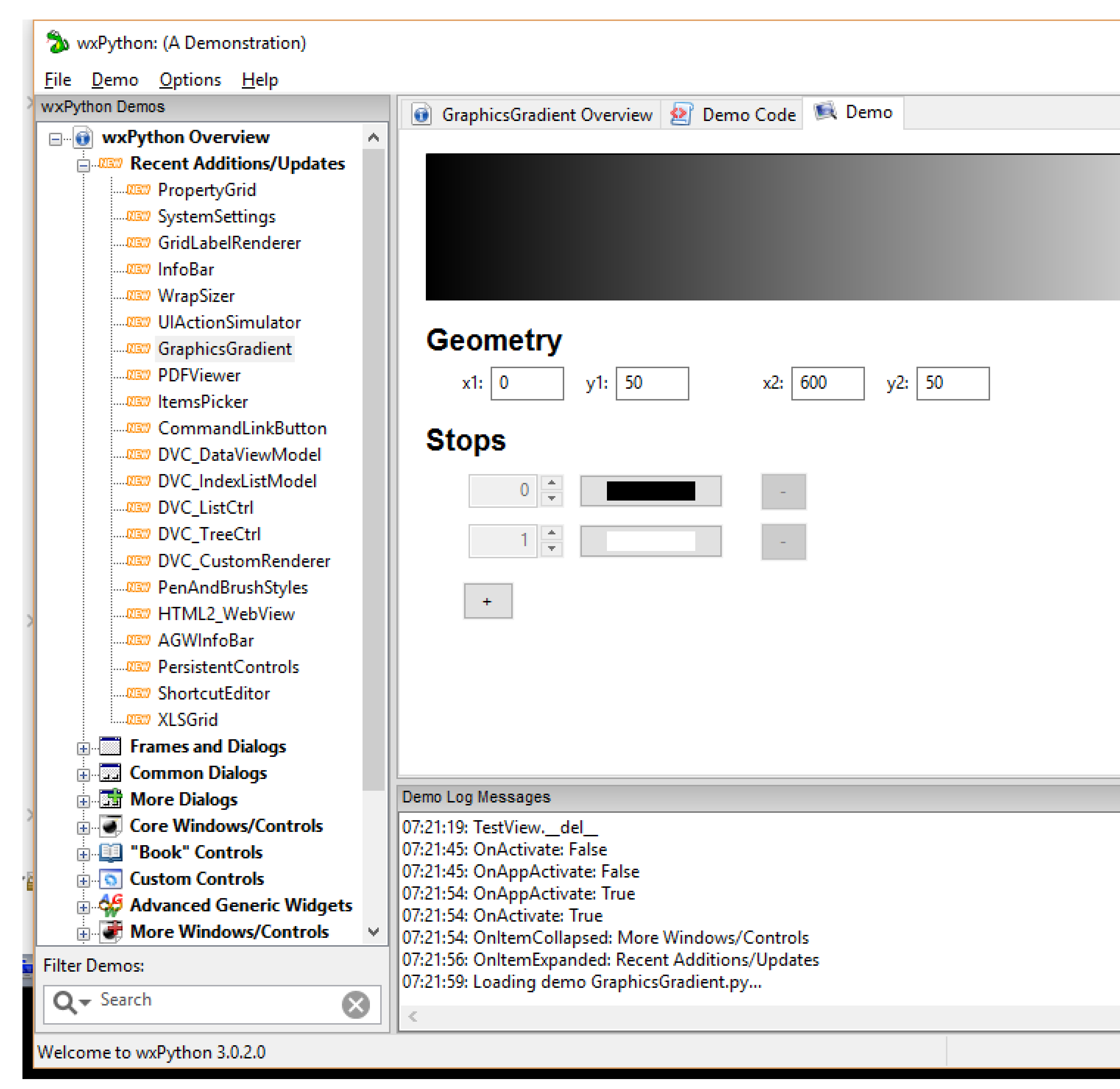
- Programadores Editores y carcasas : corteza , crustslices , AlaCart y AlaMode , AlaModeTest
- Intérprete y magia
- Inspección: esto le permite abrir una ventana para explorar todos los componentes de la GUI de sus aplicaciones.
- Un extenso conjunto de Demos.

Demo Screenshots *en Win10*

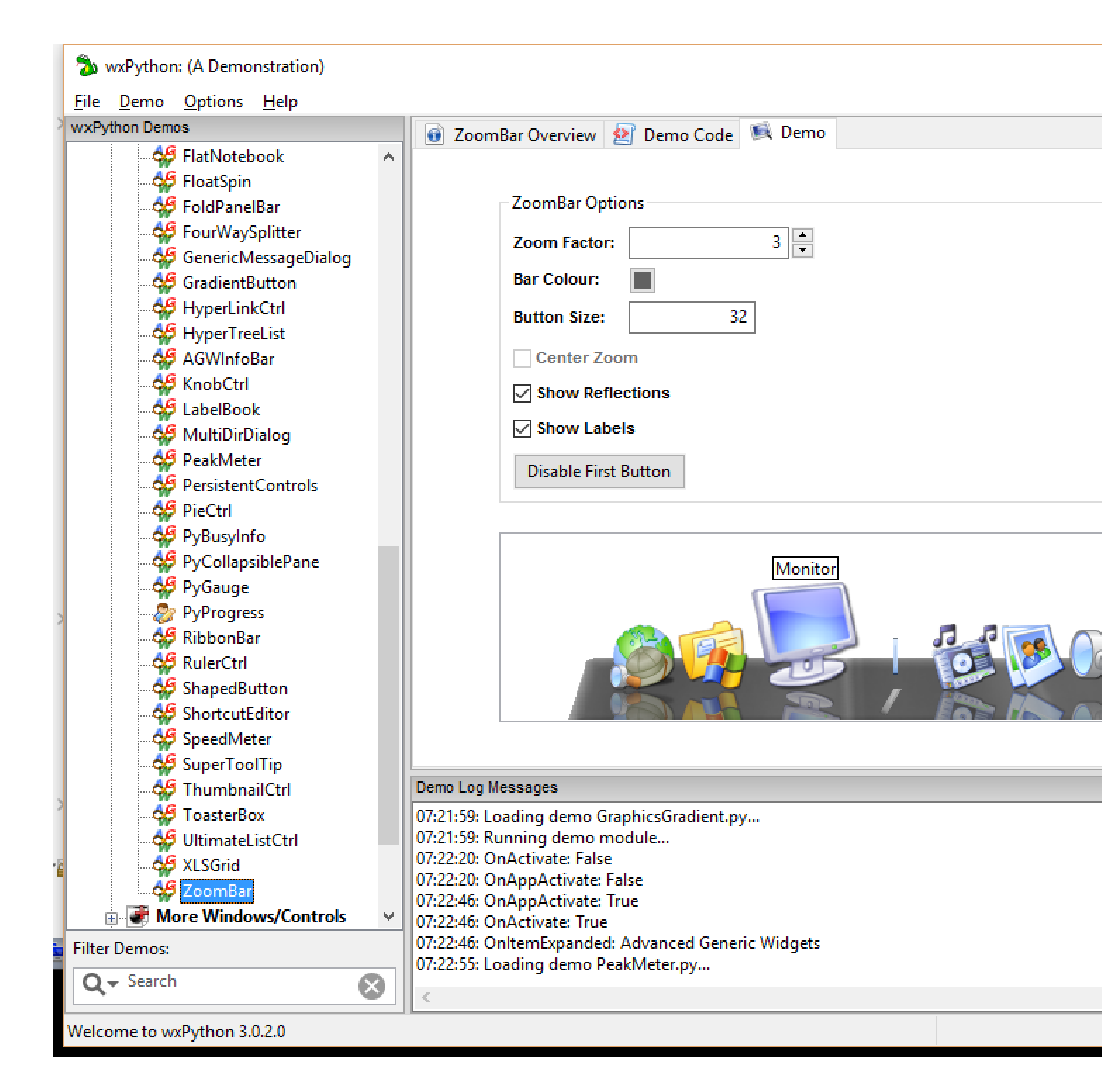
La demostración de wxPython con todas las ramas cerradas:



Una de las últimas incorporaciones:



Uno de los AGW, (Widgets genéricos avanzados):



Examples

Instalación de wxPython Phoenix

wxPython Phoenix es la última versión de wxPython, (actualmente, *septiembre de 2016* sin versión oficial). Admite Python 2 y Python 3. Puede descargar una compilación de instantáneas (es decir, una rueda de Python) para su plataforma y la versión de Python aquí .

wxPython Phoenix utiliza un mecanismo en gran medida automatizado para generar los enlaces de python para la biblioteca wxWidgets y la documentación. La documentación de Phoenix wxPython se genera específicamente para sí misma utilizando Sphinx . Esto aumenta la claridad

en oposición a la documentación en C ++ de la compilación clásica, que incluye muchas sobrecargas que no están disponibles en wxPython.

Python y **pip** deben instalarse antes de que se pueda instalar wxPython Phoenix.

Puedes usar pip para instalar la versión Phoenix de wxPython. Aquí está el método recomendado actualmente:

```
python -m pip install --no-index --find-links=http://wxpython.org/Phoenix/snapshot-builds/ --
trusted-host wxpython.org wxPython_Phoenix
```

Cuando use este comando, pip también instalará **wxWidgets** . Este comando complejo de pip probablemente se convertirá en 'pip install wxpython' cuando Phoenix se lance oficialmente.

Nota: wxPython Phoenix está actualmente en versión beta y no tiene todos los widgets que tiene la versión Classic.

Instalación de wxPython Classic

wxPython Classic es una compilación de **Python 2** de la biblioteca wxPython. La generación de los enlaces de python requiere una gran cantidad de intervenciones manuales y la documentación es simplemente la documentación de wxWidgets que contiene algunas anotaciones sobre los mecanismos de wxPython. Por lo general, hay un retraso de semanas a meses entre una nueva versión de wxWidgets y la versión correspondiente de wxPython. .

Vaya a la página de descarga en el sitio web de wxPython para ver si ya existe una versión de wxPython que pueda descargar para su plataforma.

La última versión de Classic es **3.0.2.0**

Windows

Hay instaladores para Python 2.6 y 2.7 para plataformas Windows de 32 y 64 bits en el sitio web. Solo descarga uno de estos y ejecútalos para instalarlo.

Nota: asegúrese de descargar un instalador de wxPython para el Python correcto que ha instalado. Por ejemplo, si tiene Python 2.7 de 32 bits, entonces desea un instalador de 32 bits de wxPython

Mac

Si tiene OSX **10.5 o superior** , entonces querrá descargar e instalar la versión **Cocoa** de wxPython. La versión Cocoa también es compatible con Mac de 64 bits.

Si tiene una Mac con una versión de OSX inferior a **10.5** , entonces querrá la compilación de **Carbon** .

Linux

Lo primero que debe comprobar si el administrador de paquetes de su plataforma Linux (es decir,

yum, apt-get, etc.) para ver si tiene una versión de wxPython que pueda instalar. Desafortunadamente, muchos paquetes de Linux para wxPython son para la versión 2.8.12.1 en lugar de 3.0.2.0. Si su administrador de paquetes no tiene la última versión, probablemente tendrá que compilarlo usted mismo.

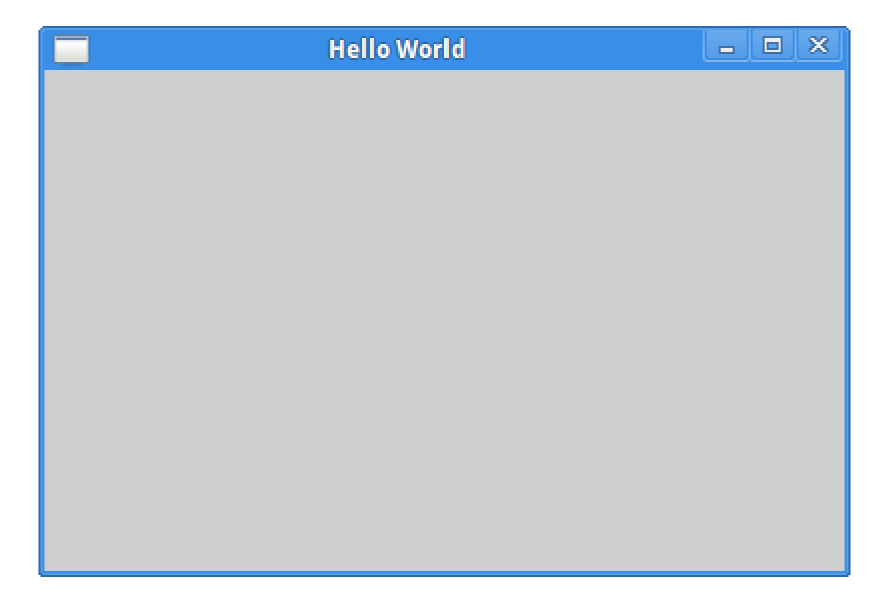
Hay instrucciones de compilación para 3.0.2.0-Classic aquí

Hola Mundo

Una forma sencilla de crear un programa **Hello World** :

```
import wx
app = wx.App(redirect=False)
frame = wx.Frame(parent=None, id=wx.ID_ANY, title='Hello World')
frame.Show()
app.MainLoop()
```

Salida:



Un ejemplo más típico sería subclase **wx.Frame** :

```
import wx

class MyFrame(wx.Frame):

    def __init__(self):
        wx.Frame.__init__(self, None, title='Hello World')
        self.Show()

if __name__ == '__main__':
    app = wx.App(redirect=False)
    frame = MyFrame()
    app.MainLoop()
```

Esto también se puede reescribir para usar el **super** de Python:

```
import wx
```

```
class MyFrame(wx.Frame):

    def __init__(self, *args, **kwargs):
        """Constructor"""
        super(MyFrame, self).__init__(*args, **kwargs)
        self.Show()

if __name__ == '__main__':
    app = wx.App(False)
    frame = MyFrame(None, title='Hello World')
    app.MainLoop()
```

¿Qué es una serie de versiones wxPython?

El proyecto wxWidgets ha adoptado el modelo de lanzamiento utilizado por el proyecto Kernel de Linux, donde existen conjuntos alternativos de lanzamientos en los que un conjunto se considera "estable" y el siguiente conjunto se considera "desarrollo". Para wxWidgets, "estable" y "desarrollo" no se refieren a errores, sino a la estabilidad de la API y la compatibilidad con versiones anteriores.

- **Estable** : Durante la duración de la serie, las API existentes no se modifican, aunque se pueden agregar nuevos métodos de clase no virtuales y otros. La compatibilidad binaria de las bibliotecas de C ++ se mantiene al no permitir ningún cambio que modifique el tamaño en memoria o el diseño de las clases y estructuras. Esto puede y, a menudo, impone limitaciones sobre qué tipo de mejoras o correcciones de errores se pueden realizar en una serie de versiones estables, sin embargo, esto solo afecta a la capa C ++ porque Python es compatible con versiones anteriores y tiene connotaciones ligeramente diferentes.

- **Desarrollo** : el objetivo principal de la serie de lanzamientos de desarrollo es agregar nuevas funciones o corregir problemas que no se pudieron corregir en una serie estable debido a problemas de compatibilidad binaria, todo en un esfuerzo por crear la siguiente serie estable. Por lo tanto, durante la duración de la serie de desarrollo, las API pueden modificarse o eliminarse según sea necesario, aunque la mayor parte del tiempo la compatibilidad de nivel de fuente de C ++ se mantiene a través de macros, funciones sobrecargadas en desuso, etc. ser incompatibilidades a nivel de fuente porque no hay sobrecarga o macros, y para admitir la nueva versión de la API a veces la versión anterior tiene que ser eliminada.

Debido a los problemas de compatibilidad binaria, la última versión de desarrollo de wxWidgets / wxPython a menudo puede tener menos fallos que la última versión de la última versión estable. Sin embargo, existe el compromiso de que las API pueden estar cambiando o evolucionando entre las versiones de la serie de desarrollo.

¿Cómo funcionan los números de versión?

Para las versiones wxPython utiliza un número de versión de 4 componentes. Si bien esto se parece mucho a cómo se usan los números de versión en otros proyectos de código abierto, hay algunas diferencias sutiles. Entonces para algunos lanzamientos **ABCD** puedes deducir lo siguiente:

1. **Serie de lanzamiento** : Los dos primeros componentes del número de versión (**AB**) representan la serie de lanzamiento, y si el componente **B** es un número par, entonces es una serie estable, si es un número impar, entonces es una serie de lanzamiento de desarrollo. Por ejemplo, 2.4, 2.6 y 2.8 son estables y la API está más o menos congelada dentro de cada serie, y 2.3, 2.5 y 2.7 son desarrollo y la API y la funcionalidad pueden cambiar o evolucionar según sea necesario.

Debido a esto, puede haber cambios bastante grandes entre una serie estable y la siguiente (por ejemplo, 2.4 a 2.6) y esto a menudo arroja a la gente a la basura porque en otros proyectos, los cambios de ese magnituto habrían provocado el cambio del primer componente del número de versión. En su lugar, debe pensar en la combinación de **AB** como el número principal de la versión.

2. **Número de versión** : El tercer componente del número de versión (C) representa una de las versiones de una serie de versiones. Por ejemplo, 2.5.0, 2.5.1, 2.5.2, 2.5.3 ... son todas las versiones de la serie 2.5. (Y dado que en este caso es una serie de desarrollo, entonces la API y la funcionalidad de 2.5.3 han evolucionado para ser diferentes en algunos lugares de lo que era en 2.5.0). Las versiones de C ++ wxWidgets generalmente se detienen aquí y solo se hacen las versiones ABC.

3. Número de versión preliminar o versión de wxPython: el cuarto componente del número de versión (D) se usa para representar una versión secundaria o versiones incrementales entre las versiones oficiales de wxWidgets. Estas versiones incluyen correcciones para errores de wxWidgets que wxPython puede haber expuesto, o mejoras menores que son importantes para wxPython. Esta no es una instantánea de wxWidgets arbitraria, sino una versión probada del código con correcciones y mejoras que aún no están disponibles en wxWidgets, excepto en el repositorio de código fuente.

Fuente: https://wiki.wxpython.org/ReleaseSeries

Lea Empezando con wxpython en línea: https://riptutorial.com/es/wxpython/topic/6690/empezando-con-wxpython

Capítulo 2: Arrastrar y soltar

Introducción

wxPython proporciona varios tipos diferentes de arrastrar y soltar. Puede tener uno de los siguientes tipos: `wx.FileDropTarget` , `wx.TextDropTarget` o `wx.PyDropTarget` .

Los dos primeros son bastante autoexplicativos. El último, wx.PyDropTarget, es solo un envoltorio suelto alrededor de wx.DropTarget. Agrega un par de métodos adicionales de conveniencia que el simple wx.DropTarget no tiene. Comenzaremos con un ejemplo de wx.FileDropTarget.

Examples

FileDropTarget

```
import wx


class MyFileDropTarget(wx.FileDropTarget):
    """"""

    def __init__(self, window):
        """Constructor"""
        wx.FileDropTarget.__init__(self)
        self.window = window

    def OnDropFiles(self, x, y, filenames):
        """
        When files are dropped, write where they were dropped and then
        the file paths themselves
        """
        self.window.SetInsertionPointEnd()
        self.window.updateText("\n%d file(s) dropped at %d,%d:\n" %
                               (len(filenames), x, y))
        for filepath in filenames:
            self.window.updateText(filepath + '\n')

        return True


class DnDPanel(wx.Panel):
    """"""

    def __init__(self, parent):
        """Constructor"""
        wx.Panel.__init__(self, parent=parent)

        file_drop_target = MyFileDropTarget(self)
        lbl = wx.StaticText(self, label="Drag some files here:")
        self.fileTextCtrl = wx.TextCtrl(self,
                                        style=wx.TE_MULTILINE|wx.HSCROLL|wx.TE_READONLY)
        self.fileTextCtrl.SetDropTarget(file_drop_target)
```

```
        sizer = wx.BoxSizer(wx.VERTICAL)
        sizer.Add(lbl, 0, wx.ALL, 5)
        sizer.Add(self.fileTextCtrl, 1, wx.EXPAND|wx.ALL, 5)
        self.SetSizer(sizer)

    def SetInsertionPointEnd(self):
        """
        Put insertion point at end of text control to prevent overwriting
        """
        self.fileTextCtrl.SetInsertionPointEnd()

    def updateText(self, text):
        """
        Write text to the text control
        """
        self.fileTextCtrl.WriteText(text)


class DnDFrame(wx.Frame):
    """"""

    def __init__(self):
        """Constructor"""
        wx.Frame.__init__(self, parent=None, title="DnD Tutorial")
        panel = DnDPanel(self)
        self.Show()


if __name__ == "__main__":
    app = wx.App(False)
    frame = DnDFrame()
    app.MainLoop()
```

TextDropTarget

```
import wx


class MyTextDropTarget(wx.TextDropTarget):

    def __init__(self, textctrl):
        wx.TextDropTarget.__init__(self)
        self.textctrl = textctrl

    def OnDropText(self, x, y, text):
        self.textctrl.WriteText("(%d, %d)\n%s\n" % (x, y, text))
        return True

    def OnDragOver(self, x, y, d):
        return wx.DragCopy


class DnDPanel(wx.Panel):
    """"""

    def __init__(self, parent):
        """Constructor"""
        wx.Panel.__init__(self, parent=parent)
```

```
        lbl = wx.StaticText(self, label="Drag some text here:")
        self.myTextCtrl = wx.TextCtrl(
            self, style=wx.TE_MULTILINE|wx.HSCROLL|wx.TE_READONLY)
        text_dt = MyTextDropTarget(self.myTextCtrl)
        self.myTextCtrl.SetDropTarget(text_dt)

        sizer = wx.BoxSizer(wx.VERTICAL)
        sizer.Add(self.myTextCtrl, 1, wx.EXPAND)
        self.SetSizer(sizer)

    def WriteText(self, text):
        self.text.WriteText(text)


class DnDFrame(wx.Frame):
    """"""

    def __init__(self):
        """Constructor"""
        wx.Frame.__init__(
            self, parent=None, title="DnD Text Tutorial")
        panel = DnDPanel(self)
        self.Show()


if __name__ == "__main__":
    app = wx.App(False)
    frame = DnDFrame()
    app.MainLoop()
```

PyDropTarget

```
import  wx


class MyURLDropTarget(wx.PyDropTarget):

    def __init__(self, window):
        wx.PyDropTarget.__init__(self)
        self.window = window

        self.data = wx.URLDataObject();
        self.SetDataObject(self.data)

    def OnDragOver(self, x, y, d):
        return wx.DragLink

    def OnData(self, x, y, d):
        if not self.GetData():
            return wx.DragNone

        url = self.data.GetURL()
        self.window.AppendText(url + "\n")

        return d


class DnDPanel(wx.Panel):
```

```
    """"""

    def __init__(self, parent):
        """Constructor"""
        wx.Panel.__init__(self, parent=parent)
        font = wx.Font(12, wx.SWISS, wx.NORMAL, wx.BOLD, False)

        # create and setup first set of widgets
        lbl = wx.StaticText(self,
                            label="Drag some URLS from your browser here:")
        lbl.SetFont(font)
        self.dropText = wx.TextCtrl(
            self, size=(200,200),
            style=wx.TE_MULTILINE|wx.HSCROLL|wx.TE_READONLY)
        dt = MyURLDropTarget(self.dropText)
        self.dropText.SetDropTarget(dt)
        firstSizer = self.addWidgetsToSizer([lbl, self.dropText])

        # create and setup second set of widgets
        lbl = wx.StaticText(self, label="Drag this URL to your browser:")
        lbl.SetFont(font)
        self.draggableURLText = wx.TextCtrl(self,
                                            value="http://www.mousevspython.com")
        self.draggableURLText.Bind(wx.EVT_MOTION, self.OnStartDrag)
        secondSizer = self.addWidgetsToSizer([lbl, self.draggableURLText])

        # Add sizers to main sizer
        mainSizer = wx.BoxSizer(wx.VERTICAL)
        mainSizer.Add(firstSizer, 0, wx.EXPAND)
        mainSizer.Add(secondSizer, 0, wx.EXPAND)
        self.SetSizer(mainSizer)

    def addWidgetsToSizer(self, widgets):
        """
        Returns a sizer full of widgets
        """
        sizer = wx.BoxSizer(wx.HORIZONTAL)
        for widget in widgets:
            if isinstance(widget, wx.TextCtrl):
                sizer.Add(widget, 1, wx.EXPAND|wx.ALL, 5)
            else:
                sizer.Add(widget, 0, wx.ALL, 5)
        return sizer

    def OnStartDrag(self, evt):
        """"""
        if evt.Dragging():
            url = self.draggableURLText.GetValue()
            data = wx.URLDataObject()
            data.SetURL(url)

            dropSource = wx.DropSource(self.draggableURLText)
            dropSource.SetData(data)
            result = dropSource.DoDragDrop()


class DnDFrame(wx.Frame):
    """"""

    def __init__(self):
        """Constructor"""
```

```
        wx.Frame.__init__(self, parent=None,
                          title="DnD URL Tutorial", size=(800,600))
        panel = DnDPanel(self)
        self.Show()


if __name__ == "__main__":
    app = wx.App(False)
    frame = DnDFrame()
    app.MainLoop()
```

Lea Arrastrar y soltar en línea: https://riptutorial.com/es/wxpython/topic/9709/arrastrar-y-soltar

Creditos

S. No	Capítulos	Contributors
1	Empezando con wxpython	4444, Boštjan Mejak, Community, Mike Driscoll, Steve Barnes
2	Arrastrar y soltar	Mike Driscoll